Microsoft Knowledge Base Article - 285339

HOWTO: Create a RealTimeData Server for Excel 2002

The information in this article applies to:

- Microsoft Excel 2002
- Microsoft Visual Basic Professional Edition for Windows 6.0

This article was previously published under Q285339

SUMMARY

Microsoft Excel 2002 provides a new worksheet function, RTD, that allows you to call a Component Object Model (COM) Automatio real-time. This article describes how to use Visual Basic to create a RealTimeData Server for use with Excel's RTD function.

MORE INFORMATION

The RTD worksheet function has the following syntax:

=RTD(*ProgID,Server,String1,*[String2],...)

The first argument, *ProgID*, represents the Programmatic Identifier (ProgID) of the RealTimeData server. The *Server* argument ind RealTimeData server is run; this argument can be a null string or omitted if the RealTimeData server is to run locally. The remainin send to the RealTimeData server; each unique combination of these parameters represents one "topic," which has an associated "t example, the following illustrates calls to the RTD server that would result in three separate topic ids:

```
=RTD("ExcelRTD.RTDFunctions",,"AAA", "10")

=RTD("ExcelRTD.RTDFunctions",,"AAA", "5")

=RTD("ExcelRTD.RTDFunctions",,"aaa", "5")
```

In order for a COM Automation Server to be a RealTimeData Server for use with Excel's RTD function, it must implement the **IRTD** all of the methods of **IRTDServer**:

ServerStart

Called when Excel requests the first RTD topic for the server. **ServerStart** should return a 1 on success, and a negative value **ServerStart** method is a callback object that the RealTimeData server uses to notify Excel when it should gather updates fron

ServerTerminate

Called when Excel no longer requires RTD topics from the RealTimeData server.

ConnectData

Called whenever Excel requests a new RTD topic from the RealTimeData server.

DisconnectData

Called whenever Excel no longer requires a specific topic.

HeartBeat

Called by Excel if a given interval has elapsed since the last time Excel was notified of updates from the RealTimeData server.

RefreshData

Called when Excel is requesting a refresh on topics. **RefreshData** is called after the server notifies Excel that updates exist, ar along with the topic id and value for each topic.

Create a Sample RealTimeData Server

The following sample demonstrates how to create and use a RealTimeData server with Microsoft Excel 2002. This server simply prc seconds on a worksheet. The server accepts up to two topic strings. The first topic string can be AAA, BBB, and CCC; any other top returns #VALUE! to the RTD function. The second string is a numeric value that represents how the return value should be increme increment value defaults to 1. If the second string is not numeric, the server returns #NUM! to the RTD function.

1. Start a new ActiveX DLL project in Visual Basic.
2. On the **Project** menu, click **References**, select **Microsoft Excel 10.0 Object Library**, and then click **OK**.
3. On the **Project** menu, click **Project1 Properties**. Change the **Project Name** to **ExcelRTD**, and then click **OK**.
4. Change the **Name** property of the class module **Class1** to **RTDFunctions**. Add the following code to **RTDFunctions**:

```
Option Explicit

Implements IRtdServer  'Interface allows Excel to contact this RealTimeData server

Private m_colTopics As Collection

Private Function IRtdServer_ConnectData(ByVal TopicID As Long, Strings() As Variant, G
    '** ConnectData is called whenever a new RTD topic is requested

    'Create a new topic class with the given TopicId and string and add it to the
    'm_colTopics collection
    Dim oTopic As New Topic
    m_colTopics.Add oTopic, CStr(TopicID)
    oTopic.TopicID = TopicID
    oTopic.TopicString = Strings(0)
    If UBound(Strings) >= 1 Then oTopic.SetIncrement Strings(1)

    'For this example, the initial value for a new topic is always 0
    IRtdServer_ConnectData = oTopic.TopicValue

    Debug.Print "ConnectData", TopicID
End Function

Private Sub IRtdServer_DisconnectData(ByVal TopicID As Long)
   '** DisconnectData is called whenever a specific topic is not longer needed

   'Remove the topic from the collection
   m_colTopics.Remove CStr(TopicID)

   Debug.Print "DisconnectData", TopicID
End Sub

Private Function IRtdServer_Heartbeat() As Long
    '** Called by Excel if the heartbeat interval has elapsed since the last time
    '   Excel was called with UpdateNotify.
    Debug.Print "HeartBeat"
End Function

Private Function IRtdServer_RefreshData(TopicCount As Long) As Variant()
    '** Called when Excel is requesting a refresh on topics. RefreshData will be calle
    '   after an UpdateNotify has been issued by the server. This event should:
    '   - supply a value for TopicCount (number of topics to update)
    '   - return a two dimensional variant array containing the topic ids and the
    '     new values of each.

    Dim oTopic As Topic, n As Integer
    ReDim aUpdates(0 To 1, 0 To m_colTopics.Count - 1) As Variant
    For Each oTopic In m_colTopics
        oTopic.Update
        aUpdates(0, n) = oTopic.TopicID
        aUpdates(1, n) = oTopic.TopicValue
        n = n + 1
```

```
    Next
    TopicCount = m_colTopics.Count
    IRtdServer_RefreshData = aUpdates

   Debug.Print "RefreshData", TopicCount & " topics updated"
End Function

Private Function IRtdServer_ServerStart(ByVal CallbackObject As Excel.IRTDUpdateEvent)
    '** ServerStart is called when the first RTD topic is requested

    Set oCallBack = CallbackObject
    Set m_colTopics = New Collection
    g_TimerID = SetTimer(0, 0, TIMER_INTERVAL, AddressOf TimerCallback)
    If g_TimerID > 0 Then IRtdServer_ServerStart = 1       'Any value <1 indicates fai

    Debug.Print "ServerStart"
End Function

Private Sub IRtdServer_ServerTerminate()
    '** ServerTerminate is called when no more topics are needed by Excel.

    KillTimer 0, g_TimerID

    '** Cleanup any remaining topics. This is done here since
    '   IRtdServer_DisconnectData is only called if a topic is disconnected
    '   while the book is open. Items left in the collection when we terminate
    '   are those topics left running when the workbook was closed.

    Dim oTopic As Topic
    For Each oTopic In m_colTopics
        m_colTopics.Remove CStr(oTopic.TopicID)
        Set oTopic = Nothing
    Next

    Debug.Print "ServerTerminate"

End Sub
```

5. On the **Project** menu, click **Add Class Module**. Change the class module **Name** property to **Topic** and change the **Insta** code to the **Topic** class module:

```
Option Explicit

Private m_TopicID As Long
Private m_TopicString As String
Private m_Value As Variant
Private m_IncrementVal As Long

Private Sub Class_Initialize()
    m_Value = 0
    m_IncrementVal = 1
End Sub

Friend Property Let TopicID(ID As Long)
    m_TopicID = ID
End Property

Friend Property Get TopicID() As Long
    TopicID = m_TopicID
End Property
```

```
Friend Property Let TopicString(s As String)
    s = UCase(s)
    If s = "AAA" Or s = "BBB" Or s = "CCC" Then
        m_TopicString = s
    Else
        m_Value = CVErr(xlErrValue) 'Return #VALUE if not one of the listed topics
    End If
End Property

Friend Sub Update()
    On Error Resume Next 'the next operation will fail if m_Value is an error (like #N
    m_Value = m_Value + m_IncrementVal
End Sub

Friend Sub SetIncrement(v As Variant)
    On Error Resume Next
    m_IncrementVal = CLng(v)
    If Err <> 0 Then
        m_Value = CVErr(xlErrNum) 'Return #NUM if Increment value is not numeric
    End If
End Sub

Friend Property Get TopicValue() As Variant
    If Not (IsError(m_Value)) Then
        TopicValue = m_TopicString & ": " & m_Value
    Else
        TopicValue = m_Value
    End If
End Property
```

6. On the **Project** menu, select **Add Module**. Add the following code to the new module:

```
Public Declare Function SetTimer Lib "user32" (ByVal hWnd As Long, _
ByVal nIDEvent As Long, ByVal uElapse As Long, ByVal lpTimerFunc As Long) As Long

Public Declare Function KillTimer Lib "user32" (ByVal hWnd As Long, ByVal nIDEvent As

Public Const TIMER_INTERVAL = 5000
Public oCallBack As Excel.IRTDUpdateEvent
Public g_TimerID As Long

Public Sub TimerCallback(ByVal hWnd As Long, ByVal uMsg As Long, ByVal idEvent As Long
    oCallBack.UpdateNotify
End Sub
```

7. On the **File** menu, click **Make ExcelRTD.dll** to build the component.

Use the RTD Server in Excel

1. Start a new workbook in Microsoft Excel.
2. In cell A1, enter the following formula, and then press the ENTER key:

   =RTD("ExcelRTD.RTDFunctions",,"AAA", 5)

   The initial return value is "AAA: 0". After five seconds, the value updates to "AAA: 10" and after 10 seconds, the value upda
3. In cell A2, enter the following formula and press ENTER:

   =RTD("ExcelRTD.RTDFunctions",,"BBB", 3)

   The initial return value is "BBB: 0". Every five seconds the cell value increments by 3.

4. In cell A3, enter the following formula and press ENTER:

   =RTD("ExcelRTD.RTDFunctions",,"AAA", 5)

   The initial return value matches the contents of cell A1 because this is the same "topic" that is used in A1.

5. In cell A4, enter the following formula and press Enter:

   =RTD("ExcelRTD.RTDFunctions",,"AAA", 10)

   The initial return value is "AAA: 0." Every five seconds the cell value increments as do the other cells. Note that the return v A3 because the combination of parameters passed to the server is different.

For this illustration, the RTD server was compiled and Excel was using the run-time version of the component. For debugging purp Basic IDE.

To run in debug mode:

1. Quit Microsoft Excel and switch to the project in Visual Basic.
2. Press F5 to start the component. If the **Project Properties** dialog box appears, click **OK** to select the default option of **Wa**
3. Make sure that the Immediate window in Visual Basic is displayed. As you enter formulas in the cells and as the cells are up Immediate window in Visual Basic to see which actions are triggering the different events.

Note Regarding the DisconnectData Event

While Excel is a subscriber to your RTD server, it triggers the **DisconnectData** event when it no longer needs a topic (for example a cell). However, Excel does not call **DisconnectData** on each topic for the RTD server when the workbook is closed or Excel quits When you are creating an RTD server, you should code for any necessary clean-up of topics or other objects when the **ServerTer**

(c) Microsoft Corporation 2001, All Rights Reserved. Contributions by Lori B. Turner, Microsoft Corporation.

REFERENCES

For additional information, click the article number below to view the article in the Microsoft Knowledge Base:

284883 PRB: RTD Server Does Not Send Update Notifications to Multiple Excel Instances

Last Reviewed: 8/27/2002

Keywords: kbAutomation kbDSupport kbhowto KB285339

Send Print Help

© 2003 Microsoft Corporation. All rights reserved. Terms of use Security & Privacy Accessibility